from the subclasses of BusinessService class 48 may be similarly translated to form an output message.

In the presently preferred embodiments, ApiService class 42 utilizes Message class 44 and Field class 46 as wrappers of a document object model (DOM) class to create input messages and output messages. Message class 44 and Field class 46 are used as wrappers to simplify creation and handling of DOM documents in the DOM class. A wrapper is a class that provides changed/enhanced interface and functionality of another class (a wrapped class). Wrappers serve as decoupling mechanisms by allowing changes to the wrapped class (the DOM class) while maintaining the interface/functionality expected by users of the class that is the wrapper (Message class 44 and Field class 46).

The DOM class includes a DOM API operating in a well-known manner. In general, the DOM API is a standardized document object model with a set of interfaces capable of describing an abstract structure for a document such as, for example, an XML document. An instance of the DOM document class is used by the DOM API to create a DOM document with a structure in the form of a virtual tree. The virtual tree is well known and includes element nodes with each element node representing an element within the document. Each of the element nodes is tagged with the same tag associated with the element in the document. In addition, each of the element nodes may include a first child that is a text node containing the element. Further, element nodes may also include a second child that is an attribute node containing any attributes of the tag.

Each element node, text node and attribute node are collectively defined to be a field where the tag is the field name, the text node contains the unit of data associated with the tag, and the attribute node contains attribute names and attribute values of the tag. The DOM API includes a DOM parser capable of generating the virtual tree and randomly accessing the fields within the virtual tree to edit, insert, delete and rearrange the fields.

In the presently preferred embodiments, requests in Servlet Request Format are translated to input messages represented as a first DOM document. In addition, responses from the back-end systems layer 18 are used to generate output messages

represented as a second DOM document. The DOM API operating in conjunction with Message class 44 and Field class 46 creates the first and second DOM documents in an XML structure.

Message class 44 and Field class 46 reduce coding complexity and streamline processing for input and output messages passed between a servlet (ApiService class 42) and custom application code (subclasses of Business Service class 48). Message class 44 operates as a wrapper to restrict manipulation of the first and second DOM documents to what is necessary for operation within the business services layer 16. Field class 46 similarly restricts manipulation of the element nodes of the first and second DOM documents. Restriction of the first and second DOM documents, and corresponding element nodes, limits the full manipulative capability typically available for XML documents. Limitation of the manipulative capability within the DOM class provides an easier to use abstraction for programming while providing sufficient functionality for development of business services applications using the business services layer 16.

In one embodiment, Message class 44 is a wrapper for well-known classes within the DOM class called Document class, DOM Element class and associated ProcessingInstruction classes. In this embodiment, Message class 44 allows for more transparent updates of the DOM parser to accommodate changes in the input and output messages. In addition, Message class 44 includes convenience functions that may be utilized in generating XML text output. As described later, and detailed in the computer program listing appendix filed herewith, the convenience functions are the combination of often used method invocations into a single method to improve productivity and reduce the skill requirement of the developer. Further, Message class 44 includes functionality to name the first and second DOM documents, create the element nodes and populate the corresponding text nodes. The text nodes are populated with request parameters contained in the requests and data contained in the responses. In another embodiment, Message class 44 also includes validation functions to validate the format of data and the types of data present in the requests and responses.

Field class 46 of one embodiment is a wrapper of a well-known DOM setAttribute method within the DOM Element class. In this embodiment, Field class

46 provides simplified access methods to the text node and the attribute node of a specified element node in the first and second DOM documents. More specifically, Field class 46 returns the contents of a specified text node as a function of specification of a datatype. In addition, Field class 46 sets the attributes on a specified attribute node as a function of the datatype.

In one embodiment, the datatypes of the text and attribute nodes may be specified as short integer, long integer, Boolean or string. Short integer is 16-bit signed two's complement integers. Long integer is 64-bit signed two's complement integers. Boolean includes 8- bits of space and 1 bit of data indicating true/false. String is a series of characters referred to as a string literal. In addition, fields may be specified with the datatype of group. Group is a field containing additional fields with datatypes. Groups may also contain additional groups. In other embodiments, fewer or additional datatypes and/or formats may be specified.

A FldTypes class (not shown) provides definition of the datatypes for data in the fields of the first and second DOM documents wrapped by Message class 44. In other embodiments, the FldTypes class also performs validation of data received in the fields by confirming the nodes within the fields include data that is the defined datatype. In yet another embodiment, Field class 46 also performs validation that the contents of a text node or attribute node within a corresponding field are the datatype expected.

MESSAGEDEFINITION class 50 of one embodiment provides meta-information in the form of a listing of valid fields common to all messages handled by Message class 44. Meta-information describes the structure and layout of the messages and is useful in debugging and validation. The meta-information greatly simplifies repetitive fields since the repetitive fields need only be declared once in MESSAGEDEFINITION class 50.

The valid fields are specified for fields expected in every request and fields expected in every response. In one embodiment, MESSAGEDEFINITION class 50 includes a first subclass defining common structures for input messages generated from requests and a second subclass defining common structures for output messages generated from responses. Each field in MESSAGEDEFINITION class 50 is preferably described using a type, a first field name referred to as shortname and a